needed to manage/present billing and authorization requirements. Tables or records in the repository **1610** may contain a listing of methods **1803** of all web services registered with the web services management system **1600**. Methods **1803** of different web services may be organized together into a bundle **1802**.

[**0136**] Packages **1801** organize bundles **1802** into default billing/authorization plans to which client applications **15** may subscribe. The same bundle can reside in different packages **1801**, with different billing/authorization specifications. In the example shown in **FIG. 18**, the basic bundle is contained in both packages **1801**.

[**0137**] A client application **15** may only subscribe to one package **1801**. This limitation advantageously reduces the logistics since allowing a client application to subscribe to more than one package **1801** may create a potential scenario of the two packages **1801** both containing the same bundle **1802** (or even different bundles **1802** with the same methods) but associated with a different billing/authorization plan. For example, the basic bundle in Package 1 is priced at $1 per use and is limited to use on weekdays. The basic bundle in Package 2 contains the same web services methods the basic bundle in Package 1. However, there are no limitations to availability for the basic bundle in Package 2 and therefore, a price of $2 per use is assigned to Package 2.

[**0138**] In one example of a billing scheme, a web service administrator can override the default billing/authorization plan specified in the package **1801** to which a client application **15** subscribes. Advantageously, this allows for further flexibility for billing client applications **15**.

[**0139**] Billing and/or authorization schemes may be created to vary depending on the time of day, day of the week, week of the month, etc. Alternatively, billing/authorization may expire after a set amount of time has passed since the package **1801** was assigned to the client application. For example, a client application may sign up for a trial package of web services that allows the client application to use the services for a trial period, like a month, after which access will expire. Thus, the system **1600** may be designed to be aware of time and temporal cycles using time stamps as to when the client application signed up, the server clock, and other means of measuring time available to the server upon which the web services management system **1600** is implemented.

[**0140**] The logging and metering module **1640** and authentication and authorization module **1620** modules may employ a plug-in architecture which allows new components to be easily added without changing the core architecture. There are several ways in which this plug-in archetecture can be accomplished. One example provides that the core architecture would search a particular location on the server, in which the web services management system **1600** is implement, for the existence of billing/authorization schemes. Since the architecture can be designed to always search the same location, additional schemes can be added. For example, other bundles **1802** may be added to a package **1801**, or other web services methods **1803** may be added to a bundle **1802**.

[**0141**] The determination of a client application's permissions to use a given resource may depend on the current standing of the client application's billing data. For example, if a client application's billing type specifies that the client application has paid a certain fee which allows the client application to make X number of calls to a bundle of methods, then on the (X+1)th call that client application will not be authorized to make the call. To achieve this, the authentication and authorization module **1620** may query the logging and metering module **1640**, or the logging and metering module **1640** may set some state in the authentication and authorization module **1620**.

[**0142**] **FIG. 19** shows a method for billing and authorization of web services (**1900**) in accordance with an embodiment of the present invention. The method (**1900**) begins with providing a listing of web service functionality (i.e., a listing of packages **1801**) to which a client application **15** may subscribe (**1901**). Subscribed client applications are given access to method calls **1803** of web services contained in bundles **1802** of a package **1801** to which they are a subscriber (**1902**). Access to web services method calls **1803** contained in the bundles **1802** is metered (**1903**). The web services usage is billed pursuant to the billing scheme of the package **1801** (**1904**). The method is done (**1905**).

[**0143**] Other steps may be added to the method (**1900**), such as registering a web service with the web services management system **1600**, registering a client application **15** with the web services management system **1600**, and storing in the repository **1610** the web services and the client applications **15** which are authorized to access the web services methods **303**.

[**0144**] The web services infrastructure **501**, **1601** creates value directly to the web services provider **20**, in that it specifically addresses the need for a web service hosting architecture described above. It could similarly address this problem for any company that wants to develop and host a collection of web services **25**.

[**0145**] Of notable advantage is the fact that the web services **25** making use of the gateway module **300**, **500**, **900** do not have to be designed with its use in mind. Any SOAP service can be used with the web services infrastructure **501**. Furthermore neither the web services **25** nor the clients **15** are aware the gateway module **300**, **500**, **900**, lies between them. In this regard, the system **300**, **500**, **900** is transparent. Since the system **300**, **500**, **900** is built on the SOAP and XML standards, it will remain viable going forward as web services **25** and their usage continue to evolve.

[**0146**] The purpose and practical use of the gateway module **300**, **500**, **900**, is that it be deployed on the servers which host a company's web services **25**. The web services **25** are registered with the web service registry repository **530** and an administrator may set up user rights, billing schemes and any other infrastructure for these services.

[**0147**] The gateway module **300**, **500**, **900**, or web services management system **1600**, provides value to the web services provider **20**, in that it allows for the tailoring of functionality on a per-client application basis. The alternative would be to develop different versions of its web services **20** for different customers, thus requiring duplication of code and additional strains on disk space, to maintain multiple copies of similar code. The gateway module **300**, **500**, **900**, provides the framework around which the web services provider **20** can track usage of the web services **25** and charge accordingly, on a per-client application basis.